**FOODBOX**

*Phase 6 Project*

Sachin Davis Mundassery

Source Code:

**FrontEnd**

<app.component.html>

<div class="container-fluid px-5">

  <router-outlet></router-outlet>

</div>

<app.module.ts>

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { HttpClientModule } from '@angular/common/http'

import { AppRoutingModule } from './app-routing.module';

import { AppComponent } from './app.component';

import { HomeComponent } from './home/home.component';

import { ProductListComponent } from './product-list/product-list.component';

import { FormsModule, ReactiveFormsModule } from '@angular/forms';

import { CheckoutComponent } from './checkout/checkout.component';

import { CartComponent } from './checkout/cart/cart.component';

import { PaymentComponent } from './checkout/payment/payment.component';

import { OrderDetailsComponent } from './checkout/order-details/order-details.component';

import { LoginComponent } from './login/login.component';

import { UserPortalComponent } from './user-portal/user-portal.component';

import { ProductDetailComponent } from './product-detail/product-detail.component';

import { SignupComponent } from './signup/signup.component';

@NgModule({

  declarations: [

    AppComponent,

    HomeComponent,

    ProductListComponent,

    CheckoutComponent,

    CartComponent,

    PaymentComponent,

    OrderDetailsComponent,

    LoginComponent,

    UserPortalComponent,

    ProductDetailComponent,

    SignupComponent,

  ],

  imports: [

    BrowserModule,

    AppRoutingModule,

    HttpClientModule,

    ReactiveFormsModule,

    FormsModule

  ],

  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }

<app.component.ts>

import { Component } from '@angular/core';

import { CartService } from './common/service/cart.service';

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css'],

  providers: [CartService]

})

export class AppComponent {

  title = 'FOODBOX';

}

<home.component.html>

<div class="row">

    <div class="col">

        <!-------------------------------------- NAVIGATION BAR  ----------------------------------------------------->

        <nav class="navbar navbar-expand-lg">

            <div class="container">

                <a class="navbar-brand" style="color: rgb(3, 6, 29); font-weight: bolder; font-size: 2.5rem; "

                    href="#">Foodbox</a>

                <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarScroll"

                    aria-controls="navbarScroll" aria-expanded="false" aria-label="Toggle navigation">

                    <span class="navbar-toggler-icon"></span>

                </button>

                <div style="width: 18rem;">

                    <ul class="navbar-nav me-auto my-2 my-lg-0">

                        <li class="nav-item">

                            <a class="nav-link active" href="#home"><span

                                    class="glyphicon glyphicon-shopping-cart"></span></a>

                        </li>

                        <li class="nav-item">

                            <h1 class="text-center" style="font-size: large;  color: rgb(212, 88, 16);">

                                Hi, {{user.username}}</h1>

                        </li>

                    </ul>

                </div>

            </div>

        </nav>

    </div>

    <div class="col-auto align-self-center">

        <a [hidden]="!user.admin" class="btn btn-primary me-1" data-bs-toggle="modal"

            data-bs-target="#staticBackdrop">Change password</a>

        <a routerLink="/checkout" class="btn btn-primary"><i class="bi bi-cart"></i> {{cart.items}}</a>

    </div>

</div>

<!-- ------------------------------------ HOME SECTION  ---------------------------------------------------

    <div class="col" section="home">

        <img src="../../assets/images/bg1.png" style="height: 35rem; width: 100%;">

    </div> -->

<div style="width: 8rem;">

    <br><br>

</div>

<!-------------------------------------- SEARCH BAR  ----------------------------------------------------->

<div class="row">

    <div class="col">

        <form>

            <div class="form-group">

                <input (ngModelChange)="updateFilter($event)" [(ngModel)]="searchText" class="form-control" type="text"

                    name="search" placeholder="Search by Product" />

            </div>

        </form>

    </div>

    <div class="col">

        <form>

            <div class="form-group">

                <input (ngModelChange)="updateFilterCuisine($event)" [(ngModel)]="searchTextCuisine"

                    class="form-control" type="text" name="cuisineFilter" placeholder="Filter by Cuisine" />

            </div>

        </form>

    </div>

    <div class="col">

        <form>

            <div class="form-group">

                <input (ngModelChange)="updateFilterPrice($event)" [(ngModel)]="searchTextPrice" class="form-control"

                    type="text" name="priceFilter" placeholder="Filter by Cost" />

            </div>

        </form>

    </div>

</div>

<br><br>

<div class="row mt-3">

    <app-product-list (newCartItemEvent)="addToCart($event)" (deleteProduct)="deleteProduct($event)"

        [admin]="user.admin" [products]="filteredProducts">

    </app-product-list>

</div>

<div class="modal fade" id="staticBackdrop" data-bs-backdrop="static" data-bs-keyboard="false" tabindex="-1"

    aria-labelledby="staticBackdropLabel" aria-hidden="true">

    <div class="modal-dialog">

        <div class="modal-content">

            <div class="modal-header">

                <h5 class="modal-title" id="staticBackdropLabel">Change password</h5>

                <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

            </div>

            <div class="modal-body">

                <div class="row g-3 align-items-center">

                    <form [formGroup]="changePassGroup" class="border p-4 rounded border-primary">

                        <div class="mb-3">

                            <label for="exampleInputEmail1" class="form-label">New password</label>

                            <input formControlName="username" type="password" class="form-control"

                                id="exampleInputEmail1" aria-describedby="emailHelp">

                        </div>

                        <div class="mb-3">

                            <label for="exampleInputPassword1" class="form-label">Confirm password</label>

                            <input formControlName="password" type="password" class="form-control"

                                id="exampleInputPassword1">

                        </div>

                        <button (click)="handleSubmit()" type="submit" class="btn btn-primary">Submit</button>

                    </form>

                </div>

            </div>

            <div class="modal-footer">

                <a href="/" class="btn btn-primary">Go back

                    home</a>

            </div>

        </div>

    </div>

</div>

<home.component.ts>

import { HttpClient, HttpHeaders } from '@angular/common/http';

import { Component, OnInit } from '@angular/core';

import { FormBuilder } from '@angular/forms';

import { Cart } from '../common/models/cart';

import { Product } from '../common/models/product';

import { User } from '../common/models/user';

import { AuthService } from '../common/service/auth.service';

import { CartService } from '../common/service/cart.service';

import { ProductService } from '../common/service/product.service';

const baseUrl = 'http://localhost:8080/api/changepass'

@Component({

  selector: 'app-home',

  templateUrl: './home.component.html',

  styleUrls: ['./home.component.css']

})

export class HomeComponent implements OnInit {

  products: Product[] = [];

  searchText: string = '';

  searchTextCuisine: string = '';

  searchTextPrice: string = '';

  filteredProducts: Product[] = [];

  user: User = { username: '', password: '', admin: false } as any;

  changePassGroup = this.formBuilder.group({

    username: this.user.username,

    password: ''

  })

  cart: Cart = { products: [], items: 0, total: 0 };

  selectedProduct: Product = null as any;

  constructor(private productService: ProductService,

    private cartService: CartService,

    private authService: AuthService,

    private formBuilder: FormBuilder,

    private httpClient: HttpClient) { }

  ngOnInit(): void {

    this.getAllProducts();

    this.getCart();

    this.user = this.authService.user;

  }

  getCart() {

    this.cartService.getCart().subscribe((cart) => {

      this.cart = cart

    });

  }

  getAllProducts() {

    this.productService.getAll().subscribe((products: any) => {

      this.filteredProducts = products;

      this.products = products;

    });

  }

  updateFilter(text: string) {

    this.filteredProducts = this.products.filter(p => p.name.toLowerCase().includes(text) || p.cuisine.toLowerCase().includes(text) || p.category.toLowerCase().includes(text))

  }

  updateFilterCuisine(textCuisine: string) {

    this.filteredProducts = this.products.filter(p => p.cuisine.toLowerCase().includes(textCuisine))

  }

  updateFilterPrice(textPrice: string) {

      var priceText: number = parseInt(textPrice);

      this.filteredProducts = this.products.filter(p => p.price<=(priceText))

  }

  addToCart(product: Product) {

    this.cartService.addToCart(product);

  }

  deleteProduct(product: Product) {

    this.productService.deleteProduct(product.id);

  }

  handleSubmit() {

    console.log(this.user.username);

    const body = { username: this.user.username, password: this.changePassGroup.value.password };

    const b64Pass = btoa(`${this.user.username}:${this.user.password}`)

    const authHeader = `Basic ${b64Pass}`

    const httpOptions = {

      headers: new HttpHeaders({

        Authorization: authHeader

      })

    }

    this.httpClient.post(baseUrl, body, httpOptions)

      .subscribe({

        next: (res) => console.log(res),

        error: (err) => console.log(err),

        complete: () => console.log('done')

      });

  }

}

<login.component.html>

<div class="back">

    <div class="row" style="color: rgb(235, 113, 13); margin-top: 3rem; padding-top: 2rem; ">

        <br><br>

        <h1 style="text-align: center;margin-top: 1rem; font-family:  Lucida Console, Courier New, monospace;">. Food

            Box .</h1>

        <br>

    </div>

    <div class="row" style="color: white; margin-top: 3rem; padding-top: 2rem;">

        <br><br>

        <h3 class="h3">Login Form</h3>

        <br><br>

    </div>

    <div class="row justify-content-center mt-3">

        <div class="col-auto">

            <form [formGroup]="loginForm" class="border p-4  " style="background-color: rgb(255, 255, 255);">

                <div class="mb-3">

                    <input formControlName="username" type="text" class="form-control" id="exampleInputEmail1"

                        placeholder="Username">

                </div>

                <div class="mb-3">

                    <input formControlName="password" type="password" class="form-control" id="exampleInputPassword1"

                        placeholder="Password">

                </div>

                <button (click)="handleSubmit()" type="submit" class="btn btn-dark" style="width: 100%;">Submit</button>

                <div [hidden]="!errorMessage" class="mt-2 text-white">

                    {{errorMessage}}

                </div>

                <div class="mb-3" style="text-align: center;">

                    <br>

                    <a routerLink="/signup"

                        style="color: white; text-decoration: none; background-color: green; padding-right: 3px;padding-left: 3px;">Signup</a>

                </div>

            </form>

        </div>

    </div>

    <div class="row" style="width: 25rem;">

        <br><br><br>

    </div>

</div>

<login.component.ts>

import { HttpClient, HttpErrorResponse, HttpHeaders, HttpResponse } from '@angular/common/http';

import { Component, OnInit } from '@angular/core';

import { FormBuilder } from '@angular/forms';

import { Router } from '@angular/router';

import { User } from '../common/models/user';

import { AuthService } from '../common/service/auth.service';

const baseUrl = "http://localhost:8080/api/login"

@Component({

  selector: 'app-login',

  templateUrl: './login.component.html',

  styleUrls: ['./login.component.css']

})

export class LoginComponent implements OnInit {

  errorMessage = ''

  loginForm = this.formBuilder.group({

    username: '',

    password: ''

  })

  constructor(

    private formBuilder: FormBuilder,

    private httpClient: HttpClient,

    private authService: AuthService,

    private router: Router

  ) { }

  ngOnInit(): void {

  }

  handleSubmit() {

    this.errorMessage = ''

    const b64Pass = btoa(`${this.loginForm.value.username}:${this.loginForm.value.password}`)

    const authHeader = `Basic ${b64Pass}`

    const httpOptions = {

      headers: new HttpHeaders({

        Authorization: authHeader

      })

    }

    this.httpClient.get<User>(baseUrl, httpOptions)

      .subscribe({

        next: res => this.authService.setUser(res),

        error: err => this.errorMessage = err.error.message,

        complete: () => this.router.navigate(['/'])

      }

      )

  }

}

<signup.component.html>

<div class="back">

    <div class="row" style="color: rgb(235, 113, 13); margin-top: 3rem; padding-top: 2rem; ">

        <br><br>

        <h1 style="text-align: center;margin-top: 1rem; font-family:  Lucida Console, Courier New, monospace;">. Food

            Box .</h1>

        <br>

    </div>

    <div class="row" style="color: rgb(255, 255, 255); margin-top: 3rem; padding-top: 2rem; ">

        <br><br>

        <h3 class="h3">Register User</h3>

        <br><br>

    </div>

    <div class="row justify-content-center mt-3">

        <div class="col-auto">

            <form [formGroup]="signupForm" class="border p-4" style="background-color: rgb(255, 255, 255);">

                <div class="mb-3">

                    <input formControlName="username" type="text" class="form-control" id="exampleInputUsername1"

                        aria-describedby="userHelp" placeholder="Username">

                </div>

                <div class="mb-3">

                    <input formControlName="password" type="password" class="form-control" id="exampleInputPassword1"

                        placeholder="Password">

                </div>

                <div class="mb-3">

                    <input formControlName="email" type="text" class="form-control" id="exampleInputEmail1"

                        aria-describedby="emailHelp" placeholder="Email">

                </div>

                <button (click)="handleSubmit()" type="submit" class="btn btn-dark"

                    style="width: 100%;">Register</button>

                <div [hidden]="!errorMessage" class="mt-2 text-danger">

                    {{errorMessage}}

                </div>

                <div class="mb-3" style="text-align: center;">

                    <br>

                    <a routerLink="/login"

                        style="color: white; text-decoration: none; background-color: green; padding-right: 3px;padding-left: 3px;">Login</a>

                </div>

            </form>

        </div>

    </div>

    <div class="row" style="width: 25rem;">

        <br><br><br>

    </div>

<signup.component.ts>

import { HttpClient, HttpErrorResponse, HttpHeaders, HttpResponse } from '@angular/common/http';

import { Component, OnInit } from '@angular/core';

import { FormBuilder } from '@angular/forms';

import { Router } from '@angular/router';

import { User } from '../common/models/user';

import { AuthService } from '../common/service/auth.service';

const baseUrl = "http://localhost:8080/api/addUser"

@Component({

  selector: 'app-signup',

  templateUrl: './signup.component.html',

  styleUrls: ['./signup.component.css']

})

export class SignupComponent implements OnInit {

  errorMessage = ''

  signupForm = this.formBuilder.group({

    username: '',

    password: '',

    email: ''

  })

  constructor(

    private formBuilder: FormBuilder,

    private httpClient: HttpClient,

    private authService: AuthService,

    private router: Router

  ) { }

  ngOnInit(): void {

  }

  handleSubmit() {

    this.errorMessage = ''

    const b64Pass = btoa(`${this.signupForm.value.username}:${this.signupForm.value.password}`)

    const authHeader = `Basic ${b64Pass}`

    const httpOptions = {

      headers: new HttpHeaders({

        Authorization: authHeader

      })

    }

    const user = {

    username: this.signupForm.value.username,

    password: this.signupForm.value.password,

    email: this.signupForm.value.email

    }

    this.httpClient.post<User>(baseUrl, user)

    .subscribe({

      next: res => this.authService.setUser(res),

      error: err => this.errorMessage = err.error.message,

      complete: () => this.router.navigate(['/login'])

    }

    )

  }

}

<productlist.component.html>

<div class="row gy-3">

    <div \*ngFor="let product of products" class="mx-auto mx-sm-0 col-auto col-sm-6 col-lg-3">

        <div class="card" style="width: 18rem;">

            <img src="../../assets/images/{{product.img}}" class="card-img-top" alt="..." style="height: 14rem;">

            <div class="card-body">

                <h5 class="card-title"><a data-bs-toggle="modal" data-bs-target="#detailView"

                        (click)="selectProduct(product)">{{product.name}}</a></h5>

                <p class="card-text">₹{{product.price}}/-</p>

                <!-- <p class="card-text">{{product.cuisine}} Cuisine</p> -->

            </div>

            <div class="card-body d-flex">

                <a href="#" (click)="newCartItemEvent.emit(product)" class="card-link btn btn-primary">Add to cart</a>

                <a [hidden]="!admin" (click)="deleteProduct.emit(product)" class="btn btn-primary ms-auto">Delete

                    product</a>

            </div>

        </div>

    </div>

</div>

<div class="modal fade" id="detailView" data-bs-backdrop="detail" data-bs-keyboard="false" tabindex="-1"

    aria-labelledby="staticBackdropLabel" aria-hidden="true">

    <div class="modal-dialog">

        <div class="modal-content">

            <div class="modal-header">

                <h5 class="modal-title" id="staticBackdropLabel">Change password</h5>

                <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

            </div>

            <div class="modal-body">

                <div class="row g-3 align-items-center justify-content-center">

                    <div class="col-auto">

                        <div class="card" style="width: 18rem;">

                            <img src="../../assets/images/{{selectedProduct.img}}" class="card-img-top" alt="...">

                            <div class="card-body">

                                <h5 class="card-title">{{selectedProduct.name}}</h5>

                                <p class="card-text">{{selectedProduct.description}}</p>

                            </div>

                            <ul class="list-group list-group-flush">

                                <li class="list-group-item">Category: {{selectedProduct.category}}</li>

                                <li class="list-group-item">Price: {{selectedProduct.price}}</li>

                                <li class="list-group-item">Company: {{selectedProduct.cuisine}}</li>

                            </ul>

                        </div>

                    </div>

                </div>

            </div>

        </div>

    </div>

</div>

<div style="width: 12rem;">

    <br><br>

</div>

<productlist.component.ts>

import { Component, EventEmitter, Input, OnInit, Output } from '@angular/core';

import { Product } from '../common/models/product';

@Component({

  selector: 'app-product-list',

  templateUrl: './product-list.component.html',

  styleUrls: ['./product-list.component.css'],

})

export class ProductListComponent implements OnInit {

  @Input() products: Product[] = [];

  @Input() admin: boolean = false;

  selectedProduct: Product = {

    id: 0,

    name: '',

    category: '',

    price: 0,

    cuisine: '',

    description: '',

    qty: 0,

    img: '',

  };

  @Output() newCartItemEvent = new EventEmitter();

  @Output() deleteProduct = new EventEmitter();

  constructor() {}

  ngOnInit(): void {}

  selectProduct(product: Product) {

    this.selectedProduct = product;

  }

}

<checkout.component.html>

<div class="row">

    <div class="col">

        <h1 class="text-center">Order Summary</h1>

    </div>

</div>

<div class="row">

    <div class="col">

        <app-cart [cart]="cart"></app-cart>

    </div>

    <div class="col-2">

        <div class="card">

            <div class="card-body" style="width: fit-content;">

                <h5 class="card-title">Proceed to buy</h5>

                <h6 class="card-subtitle mb-2 text-muted">Total no. of items: {{cart.items}}</h6>

                <p class="card-text">Total: ₹{{cart.total}}</p>

                <a routerLink="/pay" class="btn btn-primary">Pay now</a>

            </div>

        </div>

    </div>

</div>

<checkout.component.ts>

import { Component, OnInit } from '@angular/core';

import { Cart } from '../common/models/cart';

import { Product } from '../common/models/product';

import { CartService } from '../common/service/cart.service';

@Component({

  selector: 'app-checkout',

  templateUrl: './checkout.component.html',

  styleUrls: ['./checkout.component.css']

})

export class CheckoutComponent implements OnInit {

  cart: Cart = { products: [], items: 0, total: 0 }

  constructor(private cartService: CartService) { }

  ngOnInit(): void {

    this.getCart();

  }

  getCart() {

    this.cartService.getCart().subscribe((products: any) => {

      this.cart = products;

    });

  }

}

<cart.component.html>

<div \*ngFor="let product of cart.products" style="margin-left: 45%;">

    <div class="row card" style="max-width: 350px;">

        <div class="row g-0">

            <div class="col-md-3 align-self-center">

                <img src="../../assets/images/{{product.img}}" class="img-fluid rounded-start" alt="...">

            </div>

            <div class="card-body">

                <h5 class="card-title">{{product.name}}</h5>

                <p class="card-text">{{product.cuisine}} Cuisine</p>

                <p class="card-text"><small class="text-muted">Quantity: {{product.qty}}</small></p>

            </div>

        </div>

    </div>

</div>

<cart.component.ts>

import { Component, Input, OnInit } from '@angular/core';

import { Cart } from 'src/app/common/models/cart';

@Component({

  selector: 'app-cart',

  templateUrl: './cart.component.html',

  styleUrls: ['./cart.component.css']

})

export class CartComponent implements OnInit {

  @Input() cart: Cart = { products: [], items: 0, total: 0 }

  constructor() { }

  ngOnInit(): void {

  }

}

<payment.component.html>

<div class="row">

    <div class="col">

        <h3 class="text-center">Make Payment</h3>

    </div>

</div>

<div class="row">

    <div class="container" style="width: 50%; padding-top: 3rem;">

        <div class="card text-center">

            <div class="card-header">

                Please pay to confirm order...

            </div>

            <div class="card-body">

                <a class="btn btn-primary" data-bs-toggle="modal" data-bs-target="#staticBackdrop">Pay now</a>

            </div>

            <div class="card-footer text-muted">

                Please do not close or refresh this page

            </div>

        </div>

    </div>

</div>

<!-- Modal -->

<div class="modal fade" id="staticBackdrop" data-bs-backdrop="static" data-bs-keyboard="false" tabindex="-1"

    aria-labelledby="staticBackdropLabel" aria-hidden="true">

    <div class="modal-dialog">

        <div class="modal-content">

            <div class="modal-header">

                <h5 class="modal-title" id="staticBackdropLabel" style="color: rgb(95, 199, 95);">Payment successful !!

                </h5>

                <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

            </div>

            <div class="modal-body">

                <h6>Order details</h6>

                <app-cart [cart]="cart"></app-cart>

            </div>

            <div class="modal-footer">

                <a href="/" class="btn btn-primary">Go back

                    home</a>

            </div>

        </div>

    </div>

</div>

<payment.component.ts>

import { Component, OnInit } from '@angular/core';

import { Cart } from 'src/app/common/models/cart';

import { CartService } from 'src/app/common/service/cart.service';

@Component({

  selector: 'app-payment',

  templateUrl: './payment.component.html',

  styleUrls: ['./payment.component.css']

})

export class PaymentComponent implements OnInit {

  cart: Cart = { products: [], items: 0, total: 0 }

  constructor(private cartService: CartService) { }

  ngOnInit(): void {

    this.getCart();

  }

  getCart() {

    this.cartService.getCart().subscribe((products: any) => {

      this.cart = products;

    });

  }

}

**BackEnd**

1. Controller

package com.foodbox.controller;

import com.foodbox.model.Product;

import com.foodbox.model.User;

import com.foodbox.service.AuthService;

import com.foodbox.service.ProductService;

import com.foodbox.service.UserService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

import java.util.Map;

@RestController

@RequestMapping("/api")

@CrossOrigin("\*")

public class MainController {

private final ProductService productService;

private final AuthService authService;

private final UserService userService;

public MainController(@Autowired ProductService productService, AuthService authService, UserService userService) {

this.productService = productService;

this.authService = authService;

this.userService = userService;

}

@GetMapping("")

public String healthCheck() {

return "Food Box Application is running !!";

}

@GetMapping("/menu")

public List<Product> getProducts() {

return productService.getAll();

}

@PostMapping("/addMenu")

public Product addProduct(@RequestHeader String Authorization, @RequestBody Product product) {

authService.authenticate(Authorization);

return productService.add(product);

}

@PostMapping("/addUser")

public User addUser(@RequestBody User user) {

System.out.println("reached");

return userService.add(user);

}

@DeleteMapping("/menu/{id}")

public ResponseEntity<?> deleteProduct(@RequestHeader String Authorization, @PathVariable int id) {

authService.authenticate(Authorization);

productService.deleteProduct(id);

return ResponseEntity.status(HttpStatus.NO\_CONTENT).build();

}

// @GetMapping("/menu/{cuisine}")

// public List<Product> findByCuisine(@PathVariable String cuisine)

// {

// return (List<Product>) productService.getProductByCuisine(cuisine);

// }

@GetMapping("/login")

public User handleLogin(@RequestHeader String Authorization) {

return authService.authenticate(Authorization);

}

@PostMapping("/changepass")

public ResponseEntity<?> changePassword(@RequestHeader String Authorization, @RequestBody Map<String, String> rMap) {

authService.authenticate(Authorization);

this.userService.changePassword(rMap.get("username"), rMap.get("password"));

return ResponseEntity.status(HttpStatus.CREATED).build();

}

}

1. DAO

* Category

package com.foodbox.DAO;

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.stereotype.Repository;

import com.foodbox.model.Category;

import java.util.List;

@Repository

public class CategoryDAO extends AbstractDAO<Category> {

protected CategoryDAO(JdbcTemplate jdbcTemplate) {

super(jdbcTemplate);

}

@Override

public List<Category> getAll() {

return null;

}

@Override

public Category getOneById(int id) {

return jdbcTemplate.queryForObject("SELECT \* FROM category WHERE id = ?", (rs, rowNum) -> {

Category category = new Category();

category.setId(rs.getInt(1));

category.setName(rs.getString(2));

return category;

}, id);

}

@Override

public Category add(Category category) {

return null;

}

}

* Cuisine

package com.foodbox.DAO;

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.stereotype.Repository;

import com.foodbox.model.Cuisine;

import java.util.List;

@Repository

public class CuisineDAO extends AbstractDAO<Cuisine> {

protected CuisineDAO(JdbcTemplate jdbcTemplate) {

super(jdbcTemplate);

}

@Override

public List<Cuisine> getAll() {

return null;

}

@Override

public Cuisine getOneById(int id) {

return jdbcTemplate.queryForObject("SELECT \* FROM cuisine WHERE id = ?", (rs, rowNum) -> {

Cuisine cuisine = new Cuisine();

cuisine.setId(rs.getInt(1));

cuisine.setName(rs.getString(2));

return cuisine;

}, id);

}

@Override

public Cuisine add(Cuisine cuisne) {

return null;

}

}

* Product

package com.foodbox.DAO;

import com.foodbox.model.Product;

import com.foodbox.model.User;

import com.foodbox.service.CategoryService;

import com.foodbox.service.CuisineService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.dao.DuplicateKeyException;

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.stereotype.Repository;

import java.util.List;

@Repository

public class ProductDAO extends AbstractDAO<Product> {

private final CategoryService categoryService;

private final CuisineService cuisineService;

protected ProductDAO(JdbcTemplate jdbcTemplate, @Autowired CategoryService categoryService, @Autowired CuisineService cuisineService) {

super(jdbcTemplate);

this.categoryService = categoryService;

this.cuisineService = cuisineService;

}

@Override

public List<Product> getAll() {

return this.jdbcTemplate.query("SELECT \* FROM product", (rs, rowNum) -> {

Product product = new Product();

product.setId(rs.getInt(1));

product.setName(rs.getString(2));

product.setCategory(categoryService.getCategoryById(rs.getInt(3)).toString());

product.setPrice(rs.getInt(4));

product.setCuisine(cuisineService.getCuisineById(rs.getInt(5)).toString());

product.setImg(rs.getString(6));

return product;

});

}

@Override

public Product getOneById(int id) {

return null;

}

@Override

public Product add(Product product) throws DuplicateKeyException {

this.jdbcTemplate.update("INSERT INTO product VALUES (?, ?, ?, ?, ?, ?, ?)",

product.getId(), product.getName(), product.getCategory(), product.getPrice(), product.getCuisine(), product.getImg());

return product;

}

public void deleteProduct(int id) {

System.out.println(id);

this.jdbcTemplate.update("DELETE FROM product WHERE id = ?", id);

}

}

* User

package com.foodbox.DAO;

import org.springframework.dao.DuplicateKeyException;

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.stereotype.Repository;

import com.foodbox.model.Product;

import com.foodbox.model.User;

import java.util.List;

@Repository

public class UserDAO extends AbstractDAO<User> {

protected UserDAO(JdbcTemplate jdbcTemplate) {

super(jdbcTemplate);

}

@Override

public List<User> getAll() {

return null;

}

@Override

public User getOneById(int id) {

return null;

}

@Override

public User add(User userDetails) {

return null;

}

public User getUserByUsername(String username) throws Exception {

User res = this.jdbcTemplate.queryForObject("SELECT \* FROM user WHERE username = ?", (rs, rowNum) -> {

User user = new User();

user.setUsername(rs.getString(1));

user.setPassword(rs.getString(2));

user.setAdmin(rs.getInt(4) == 1);

return user;

}, username);

return res;

}

public void update(String username, String pass) {

this.jdbcTemplate.update("UPDATE user SET password = ? WHERE username = ?", pass, username);

}

public User addUser(User user) throws DuplicateKeyException {

this.jdbcTemplate.update("INSERT INTO user VALUES (?, ?, ?, ?)",

user.getUsername(), user.getPassword(), user.getEmail(), 0);

return user;

}

}

1. Model

* Category

**package** com.foodbox.model;

**public** **class** Category {

**private** **int** id;

**private** String name;

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

@Override

**public** String toString() {

**return** **this**.name;

}

}

* Cuisine

**package** com.foodbox.model;

**public** **class** Cuisine {

**private** **int** id;

**private** String name;

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

@Override

**public** String toString() {

**return** **this**.name;

}

}

* Product

**package** com.foodbox.model;

**public** **class** Product {

**private** **int** id;

**private** String name;

**private** String category;

**private** **int** price;

**private** String cuisine;

**private** String img;

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getCategory() {

**return** category;

}

**public** **void** setCategory(String category) {

**this**.category = category;

}

**public** **int** getPrice() {

**return** price;

}

**public** **void** setPrice(**int** price) {

**this**.price = price;

}

**public** String getCuisine() {

**return** cuisine;

}

**public** **void** setCuisine(String cuisine) {

**this**.cuisine = cuisine;

}

**public** String getImg() {

**return** img;

}

**public** **void** setImg(String img) {

**this**.img = img;

}

}

* User

**package** com.foodbox.model;

**public** **class** User {

**private** String username;

**private** String password;

**private** String email;

**private** **boolean** isAdmin;

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

**public** String getUsername() {

**return** username;

}

**public** **void** setUsername(String username) {

**this**.username = username;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

**public** **boolean** isAdmin() {

**return** isAdmin;

}

**public** **void** setAdmin(**boolean** admin) {

isAdmin = admin;

}

}

1. Services

* Auth

package com.foodbox.service;

import com.foodbox.DAO.UserDAO;

import com.foodbox.exception.InvalidCredentialsException;

import com.foodbox.model.User;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import javax.servlet.http.HttpServletRequest;

import java.util.Base64;

@Service

public class AuthService {

private final UserDAO userDAO;

public AuthService(@Autowired UserDAO userDAO) {

this.userDAO = userDAO;

}

private User isAuthenticated(String username, String password) throws Exception {

User user = userDAO.getUserByUsername(username);

if (user.getPassword().equals(password)) return user;

else return null;

}

public User authenticate(String authHeader) {

String base64 = authHeader.substring(6);

String userpass = new String(Base64.getDecoder().decode(base64));

String[] creds = userpass.split(":");

User auth = null;

try {

auth = isAuthenticated(creds[0], creds[1]);

} catch (Exception e) {

throw new InvalidCredentialsException();

}

if (auth == null) throw new InvalidCredentialsException();

return auth;

}

}

* Category

package com.foodbox.service;

import com.foodbox.DAO.DAO;

import com.foodbox.model.Category;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

@Service

public class CategoryService {

private final DAO<Category> categoryDAO;

public CategoryService(@Autowired DAO<Category> categoryDAO) {

this.categoryDAO = categoryDAO;

}

public Category getCategoryById(int id) {

return categoryDAO.getOneById(id);

}

}

* Cuisine

package com.foodbox.service;

import com.foodbox.DAO.CuisineDAO;

import com.foodbox.model.Cuisine;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

@Service

public class CuisineService {

@Autowired

private final CuisineDAO cuisneDAO;

public CuisineService(CuisineDAO cuisneDAO) {

this.cuisneDAO = cuisneDAO;

}

public Cuisine getCuisineById(int id) {

return this.cuisneDAO.getOneById(id);

}

}

* Product

package com.foodbox.service;

import com.foodbox.DAO.DAO;

import com.foodbox.DAO.ProductDAO;

import com.foodbox.model.Product;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class ProductService {

private final DAO<Product> productDAO;

public ProductService(@Autowired ProductDAO productDAO) {

this.productDAO = productDAO;

}

public List<Product> getAll() {

return productDAO.getAll();

}

public Product add(Product product) {

return productDAO.add(product);

}

public void deleteProduct(int id) {

((ProductDAO) this.productDAO).deleteProduct(id);

}

}

* User

package com.foodbox.service;

import com.foodbox.DAO.DAO;

import com.foodbox.DAO.UserDAO;

import com.foodbox.model.Product;

import com.foodbox.model.User;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

@Service

public class UserService {

private final DAO<User> userDAO;

public UserService(@Autowired UserDAO userDAO) {

this.userDAO = userDAO;

}

public Boolean changePassword(String username, String pass) {

((UserDAO) this.userDAO).update(username, pass);

return true;

}

public User add(User user) {

return ((UserDAO) userDAO).addUser(user);

}

}

1. Application Properties

spring.datasource.url=jdbc:mysql://localhost:3306/foodbox

spring.datasource.username=root

spring.datasource.password=root

spring.datasource.driver-class-name=com.mysql.jdbc.Driver

springdoc.swagger-ui.path=/swagger-ui.html

1. Pom

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.6.2</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.Food-Box</groupId>

<artifactId>Foof-Box</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>Kitchen Story</name>

<description>Food-Box</description>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-jdbc</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.37</version>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>